Réalisation d'un système solaire

version 1

Interface Homme-Machine : Unity

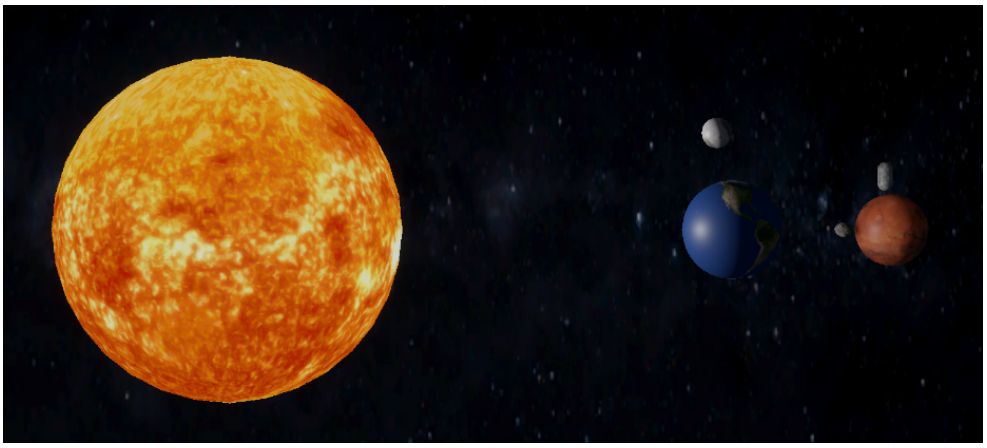
Voici les objectifs de ce sujet :

— Comprendre l'IDE `Unity`.
— Création d'un projet.
— Placer des objets dans l'environnement.
— Comprendre la hiérarchie pour différencier transformation locale et globale.
— Comprendre le canevas 2D.
— Manipuler des widgets classiques.
— Exploiter les événements pour ajouter des interactions.
— Réaliser des scripts simples pour l'animation.

1 Description générale de l'application

Dans ce TP, nous allons modéliser un système solaire qui n'est pas réaliste physiquement. Pour cela, vous devez réaliser en vous inspirant ce qui a été fait en cours en utilisant la hiérarchie, un système solaire composé des éléments suivants :

— *Soleil* qui est au centre de notre animation ;
— deux planètes (disons *Terre* et *Mars*) qui tournent autour du *Soleil* ;
— des lunes pour chaque planète. Pour information, la *Terre* possède une *Lune* et *Mars* possède 2 lunes (nommées *Phobos* et *Deimos*).



L'objectif ici est de réalisez des manipulations d'Unity, vous pouvez/devez expérimenter les points suivants :

1. Utilisez la forme primitive `Capsule` pour *Deimos*.
2. Utilisez un maillage produit avec vos mains sous Blender ou *via* un objet quelconque (format `.obj` de préférence) trouvé sur l'Internet pour remplacer *Phobos*.
3. Mettez une texture sur les objets (pas forcément celle des planètes, mais ce que vous voulez).
4. Modifiez la `"skybox"` de votre caméra (plusieurs possibilités sont permises, mais utilisez bien la documentation et vos intuitions pour le faire, sans chercher en premier lieu une solution sur le net).
5. Placez les objets *de manière hiérarchique* pour répercuter les transformations de leur parent.

Vous pouvez utiliser les ressources mises à disposition sur `UPdago`. Depuis l'interface de `Unity` : repérer l'onglet `Project` et le dossier `Assets`, dans lequel vous glissez-déposez le dossier `Images`.

2 Premier script pour animer tout cela

Je vous propose de réaliser le fameux script `JeTourne.cs` du cours, qui se contente d'appeler la rotation autour du soleil. Chercher dans la documentation `Unity`, le rôle de la fonction `FixedUpdate()`. Modifiez votre programme pour l'utiliser en conséquence comme dans le cours. Nous ferons en particulier les variables suivantes sans les lier à des boutons ou autre widget :

— Ajoutez dans votre script une variable booléenne qui détermine si l'objet en question tourne ou non.

— Placez la vitesse de rotation de votre objet paramètre.

— Surchargez les fonctions intéressantes du cycle de vie de votre objet pour en faire un affichage dans la console.

3 Interaction pour le contrôle du système solaire

Réalisez l'interface proposée ci-dessous pour le soleil :

Ensuite sur ce même modèle, réalisez l'interface pour la *Terre* et *Mars* uniquement.

Maintenant que vous avez une jolie interface, vous allez réaliser la connexion au écouteur d'événement des boutons d'animation pour contrôler si la rotation de l'astre associé est active ou non. En bref, il vous est demandé de réaliser une fonction, qui détecte les changements de valeur de la `Checkbox` pour en activer/désactiver l'animation selon la valeur du booléen. Concernant le `Slider`, nous considérons qu'il pourra prendre des valeurs de 0 à 100 et qui contrôlera la vitesse de rotation de l'objet associé.

4 Placement de caméra

Lisez la page du guide suivant : `https://docs.unity3d.com/Manual/MultipleCameras.html`. Elle explique les deux façons de contrôler l'affichage d'une caméra simplement. [1]

Pour mettre en pratique les explications du lien, nous vous proposons d'ajouter une caméra locale à la *Lune* (de la *Terre*). Ajoutez des widgets à l'interface pour changer de caméra à la volée comme indiqué dans la documentation.

Pour les explications de la seconde partie, on se propose d'ajouter une 3ème caméra en vue de haut de notre système solaire. Cette vue devra apparaître en miniature dans le bord bas gauche de la fenêtre (ou un autre bord si vous préférez).

5 Pour aller plus loin

La mécanique céleste mise en place jusqu'à présent est vraiment basique, mais offre un cadre suffisant pour l'apprentissage des animations. Nous vous proposons d'ajouter une comète cyclique (ou autre astre) qui n'est plus sous-fils du Soleil, mais bien un objet quelconque.

Pour pouvoir animer votre comète, vous allez devoir lui fournir une équation qui va traduire sa trajectoire. Pour cela, vous pouvez vous inspirer du lien suivant `https://fr.wikipedia.org/wiki/Trajectoire_d%27une_com%C3%A8te` ou d'utiliser l'équation d'un cercle autour du soleil par exemple.

1. Ce lien évoque la propriété *Depth* permettant de déterminer quelle caméra est visible à un moment donné. Dans `Unity 6`, cette propriété est renommée `Rendering > Priority`.

6 Toujours plus loin

Si vous avez tout fini, nous pouvons complexifier les traitements en regardant la documentation :

— En utilisant la fonction `LookAt` dans la classe `Transform`, ajoutez pour chaque interface un bouton `'Center View'` qui centre la vue de la caméra principale vers l'astre associé.

— Modifier votre traitement précédent pour que la caméra tourne doucement vers sa position finale pour éviter de perdre l'utilisateur ou lui donner des nausées. Pour cela, nous utiliserons une interpolation linéaire ou sphérique entre le point de vue de départ et le point de vue d'arrivée avec une vitesse constante.